

REMARKS

In an office action dated July 29, 2004, the Examiner required updating of cross references in the specification; rejected claims 1-16 under 35 U.S.C. 112, second paragraph, as indefinite; and rejected claims 1-16 under 35 U.S.C. 103(a) as being unpatentable over Cameron et al. (U.S. Patent 5,325,526).

Specification

Applicants have amended the specification to update the references to several U.S. patent applications, which have now issued.

Indefiniteness Rejections

Applicants have amended claims 1, 5 and 11 to establish antecedent basis in response to the Examiner's rejections. Specifically, in claim 1, the reference to "said multi-processor system" has been changed to "said computer system" (objection A(i)); and "said first set" has been changed to "said first processor set" (objection A(ii)). In claim 5, "disjoint sets of central processing units" has been changed to "disjoint sets of said central processing units" (objection B). In claim 11, "said first set" has been changed to "said first processor set" (objection A(v)).

Although not objected to by the Examiner, applicants have also amended claims 1 and 11 to insert the word "processing" before "capacity values" in lines 12 and 16, respectively, for clarity of antecedent basis.

With respect to objections A(iii), A(iv) and A(vi), applicants have not amended the claims and respectfully traverse the rejections. In each case, the phrase refers to a specific set which has been introduced by the previous recitations of the claim. For example, claim 1 recites a step of: "assigning each logical partition of said plurality of logical partitions to a respective set of said

plurality of processor sets...” I.e., each logical partition is assigned to one and only one processor set (although more than one logical partition might be assigned to a single processor set). The step to which the Examiner objects recites: “constraining tasks executing in a each logical partition to execute only in central processors assigned to *the processor set to which the respective logical partition is assigned...*” Since, according to the previous step, each logical partition is assigned to one and only one processor set, the use of the article “the” when applied to the phrase “processor set to which the respective logical partition is assigned” is correct. In fact, use of the article “a” would only lead to unnecessary confusion, since it might imply that a single partition is assigned to multiple processor sets at a given time, which is not the case. The objections to claims 5 and 11 are similarly met.

Prior Art

Applicants respectfully traverse the Examiner’s prior art rejections of the claims.

A brief background discussion is in order. Despite a certain superficial similarity between constructs and terminology used by applicants and those disclosed in *Cameron*, it must be pointed out first of all that *Cameron*’s system is a fundamentally different architecture from that disclosed as applicants’ preferred embodiment. *Cameron*’s system, which it calls a “multicomputer architecture”, is an array (2 or 3 dimensional) of multiple nodes, each node having a processor and independent memory. Such a system is designed for certain computationally intensive applications, which preferably execute on multiple processors in parallel. The array architecture facilitates data flow among processors engaged in processing a task in parallel. In contrast to *Cameron*’s system, applicants’ system is intended for a multiple processor computer, in which the processors share a common memory accessible to all, and each processor typically executes a single task independently of the other processors. Such processors are typically much larger and more sophisticated than the node processors of *Cameron*’s “multicomputer architecture”.

Cameron calls this a “multiple processor environment”, and emphasizes that the two architectures are fundamentally different¹

Cameron discloses that, in a prior art multicomputer architecture, a task scheduler will typically assign all the nodes to execute a particular task at any given time. Alternatively, the nodes might be divided into fixed subsets, wherein the task scheduler limits execution of any particular application to a fixed subset. In either case, processing capacity is often wasted because different applications require different numbers of processors. *Cameron* addresses this problem by using a hierarchical partitioning, in which a particular node may be assigned to different overlapping partitions at different levels of the hierarchy. A task scheduler assigns tasks to partitions according to the number of processors needed by each respective task and the partition sizes. *Cameron*’s partitions are relatively dynamic, and can be created and removed as required for execution of particular tasks. According to *Cameron*, this makes it possible to find a better fit of numbers of processors in a partition and the needs of an application.

Applicants’ invention is directed to a different problem involving a multiple processor environment which has been logically partitioned. Logical partitioning of a multiple processor environment is used to make a single computer system behave like multiple systems. This can be done for various reasons, e.g., to support multiple different operating system environments; to isolate certain processes for reasons of security; etc. In particular, logical partitioning is used to allocate resources, including processor resources, such as where different entities are being charged a fee for use of computer resources. In prior art logical partitioning systems, any particular processor is assigned to a particular logical partition, or can be shared among multiple partitions. Unfortunately, when processors are shared among multiple partitions, it is difficult to control resource allocation. I.e., changes to a first partition may affect the performance of tasks

¹ Col. 6, lines 32-45.

executing in other partitions. In accordance with applicant's preferred embodiment, each logical partition which shares a processor or pool of processors is assigned a respective processing capacity value, expressed as a number of processor equivalents. This processing capacity value is not necessarily an integer. The tasks running in a particular partition are limited to the processing capacity value of that partition. For example, if a processing capacity value of 1.5 is specified, the tasks running in the partition are limited to an equivalent of 1.5 processors. This is the equivalent of one processor running full time on behalf of the partition, and another processor running half time on behalf of the partition (although, in reality, the processing capacity tends to be distributed more or less evenly, so that if, e.g., the processor pool contains three processors, in general the partition would be limited to about half time in each of the three processors). In the preferred embodiment, the system administrator can designate each partition's capacity as capped (meaning that tasks running in that partition are not allowed to exceed the designated processor capacity, even if some of the processors in the pool are idle) or uncapped (meaning that tasks running in that partition are allowed to use otherwise idle processor capacity in excess of the designated processor capacity).

In response to all this background, it might be objected that applicants' claims do not draw a distinction between the "multicomputer architecture" of *Cameron* and a "multiple processor environment". This would certainly be significant if the claims recited no patentable distinction over *Cameron*. But there is a very significant feature which is not disclosed in *Cameron*, and which, due to the nature of *Cameron*'s "multicomputer architecture", would not have been suggested as an obvious variation or modification of *Cameron*'s partitioning mechanism. This is the use of a "processor capacity value".

Applicants' representative claim 1, as amended, recites:

1. A method for allocating processor resources in a computer system having a plurality of central processors, comprising the steps of:

defining a plurality of logical partitions of said computer system, wherein each task executing in said computer system is assigned to a respective one of said logical partitions;

defining a plurality of sets of processors;

assigning each central processor of said computer system to a respective set of said plurality of processor sets;

assigning each logical partition of said plurality of logical partitions to a respective set of said plurality of processor sets, wherein a first processor set of said plurality of processor sets has a plurality of logical partitions assigned to it;

assigning a respective processing capacity value to each of said plurality of logical partitions assigned to said first processor set, said *capacity values representing processing capacity in units equivalent to a fixed number of physical central processors*;

constraining tasks executing in each logical partition to execute only in central processors assigned to the processor set to which the respective logical partition is assigned; and

constraining tasks executing in said each logical partition assigned to said first processor set to *execute for a combined length of time equivalent to the processing capacity value* assigned to the respective logical partition. [emphasis added]

Independent claims 5 and 11, while not identical in scope, contain limitations analogous to those italicized above.

As explained above and recited in the claims, a processor capacity value expresses processor capacity in *units equivalent to a fixed number of physical processors*.² This is a key limitation of all independent claims, which is not met by *Cameron*.

The Examiner apparently reads applicants' recited "processor capacity value" on *Cameron's* "partition execution quantum". This reading is erroneous. *Cameron's* "partition

² In the preferred embodiment, the units are identical to physical processors, but this is not necessarily required by the claims. For example, the preferred embodiment assumes that all processors are identical, but in some systems a heterogeneous mixture of processors is used, some of which may be back level and of lower processing capacity. In these cases as well as in others, the unit chosen could be any of the processors, or some different unit representing a typical or standard processor. The significance of measuring processing power in units equivalent to processors is that it does not change with overall system capacity or load.

execution quantum” limits the size of a timeslice that can be allocated to a task by the dispatcher. I.e., a task, once dispatched to a group of processors, may execute no longer than the applicable “partition execution quantum”, before being swapped out (it will typically be swapped out before the end of the timeslice, as a result of a latency event, such as a need to access disk storage). The “partition execution quantum” does not directly limit the processor capacity which is available collectively to all tasks in a partition. On the contrary, once a particular task exceeds its timeslice, what happens next depends on the system load and available queued processes. The dispatcher may very well dispatch another task in the very same partition, so that the timeslice is not even a limitation on the continuous length of time a processor or group of processors may be dedicated to executing tasks in a particular partition.

It must be conceded that, in a general sense, the use of a timeslice limit will in some circumstances “constrain” the amount of processor resource allocated to a particular task, or tasks in a particular partition. However, the same could be said of almost anything in the execution environment. E.g., the number of other tasks, the priorities of other tasks, the size of memory allocations, and many other factors, will in some way affect the amount of processor resource that gets allocated to tasks in a particular partition. The point of applicants’ invention is to *minimize* the effect of all these extraneous and ungovernable factors by providing a direct mechanism for controlling the processing capacity allocated to a partition. *Cameron’s* “partition execution quantum” does not do that.

A “partition execution quantum”, as used in *Cameron*, is a timeslice, pure and simple. It does not express a processor capacity “in units equivalent to a fixed number of physical processors”, as recited in applicants’ claims, and therefore this significant claim limitation is not met.

Nor is the limitation of a “processor capacity value”, as defined in the claims, obvious over *Cameron*. As explained above, *Cameron* uses a fundamentally different architecture to solve a different problem. The problem in *Cameron* is that it is difficult to schedule tasks in an array of processors (“multicomputer architecture”), where different tasks either require or optimally use different numbers of processors. The problem is thus one of utilizing the available capacity in the most efficient manner. There is no suggestion or motivation shown in *Cameron* to limit processor capacity of a partition in the manner claimed by applicants, for that would only make the scheduling problem worse. On the contrary, as taught by *Cameron*, the partition is itself designed to be the factor controlling processor usage. *Cameron* teaches a system of hierarchical partitions which are dynamically created and destroyed to accommodate the needs of various tasks. Once a task is dispatched within a partition, it preempts the partition and uses all the processors within it exclusively. The partition prevents the task from preempting all of the processors on the system (i.e., those outside the partition), and thus allows other tasks to execute in other processors concurrently. In these circumstances, there is no demonstrated motivation or suggestion to use a separate processor capacity value to limit access to a group of processors by tasks running in a partition.

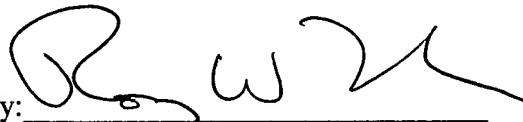
For all of the reasons stated above, applicants’ independent claims are neither taught nor suggested by *Cameron*, and the rejections thereof were erroneous.

Applicants’ dependent claims recite further limitations not disclosed or suggested by *Cameron*. Specifically, claims 2, 7 and 12 recite the ability to designate processor capacity as capped or uncapped to either prevent or allow usage of idle processor capacity. This again runs contrary to the teachings of *Cameron*, which attempts to maximize processor usage.

In view of the foregoing, applicants submit that the claims are now in condition for allowance and respectfully request reconsideration and allowance of all claims. In addition, the Examiner is encouraged to contact applicants' attorney by telephone if there are outstanding issues left to be resolved to place this case in condition for allowance.

Respectfully submitted,

WILLIAM J. ARMSTRONG, et al.

By: 

Roy W. Truelson

Registration No. 34,265

Telephone: (507) 289-6256

Docket No.: ROC920000252US1
Serial No.: 09/838,057